C++程序设计第 5 次作业　大数据 2101 班　李嘉鹏　U202115652　　2022 年 11 月 2 日

1. mutable 成员一般在什么情况下定义？mutable 成员可以同时定义为 static、const、volatile 或它们的组合吗？说明理由。

mutable 成员一般在类中存在某个 this 指向的对象不可改的成员函数（参数表后有 const）时定义，从而可以在当前对象不可修改的情况下对特定数据元素进行修改。mutable 成员不能同时定义为 static、const 或它们的组合，因为 mutable 成员依赖于 this 指针属于某一对象，但 static 或 const 类型不依赖具体的对象而存在；它可以定义为 volatile 类型，因为 volatile 具有易变属性。

2. 类的实例成员指针和静态成员指针有什么不同？在使用时为什么实例成员指针需要绑定对象（引用、对象指针)？而静态成员指针不需要绑定对象（引用、对象指针)?

类的实例成员指针依赖于类而存在，其值为成员相对类首址的偏移量，因此需要通过类（通过引用或对象指针的方式）来访问或调用，不能移动、参与运算或强制类型转换；而静态成员指针本质上就是普通指针，是整个类共用的成员，不依赖于具体的类，直接通过普通指针指向其物理地址即可访问，可以完成各种操作。

3. 分析如下定义是否正确，并指出错误原因。

```
struct A {
    char *a, b, *geta( );
    char A::*p;
    char *A::*q( );
    char *(A::*r)( );
};
int main(void) {
    A a;
    a.p = &A::a; 错误，a.p类型和A::a一致，不应取地址
    a.p = &A::b;
    a.q = &A::geta; 错误，a.q是一个函数而不是指向实例函数成员的指针，不应取地址
    a.r = a.geta; 错误，a.r 是指向实例函数成员的指针，其值应为函数成员的偏移量，而非调用 a.geta
    a.r = &a.geta; 错误，&a.geta 不是指向实例函数成员的指针（返回物理地址而非偏移量）
    a.r = &A::geta;
}
```

不正确，错误原因如上所述。

4. 分析如下定义是否正确，并指出错误原因。

```
struct A {
    static int x = 1; 错误，只有非volatile类的static const整型变量才能定义缺省值，类内的static变量需要在外部赋值
    static const int y = 2;
    static const volatile int z = 3; 错误，只有非volatile类的static const整型变量才能定义缺省值
    static volatile int w = 4; 错误，只有非volatile类的static const整型变量才能定义缺省值
    static const float u = 1.0f; 错误，只有非volatile类的static const整型变量才能定义缺省值
```

```
};
static int A::x = 11; 错误，不应再次声明static
int A::y = 22; 错误，A::y是const变量且已经用缺省值初始化，不能修改
int volatile A::z = 33; 错误，此处形式与类内定义的形式不同，应为 int const volatile A::z=33
int volatile A::w = 44;
const float A::u = 55.0f;
```
不正确，错误原因如上所述。

5. 分析如下定义是否正确，并指出错误原因。

```
class A {
    static int *j, A::*a, i[5];
public:
    int x;
    static int &k, *n;
};
int y = 0;
int A::i[5] = {1, 2, 3};
int *A::j = &y;
int A::*j = &A::x;
int A::*A::a = &A::x;
int &A::k = y;
int *A::n = &y;
```
正确。

6. 分析如下定义是否正确，并指出错误原因。

```
class A {
    int a;
    static friend int f( );
    friend int g( );
public:
    friend int A( ); 错误，析构函数不可能返回int，并且友元只能修饰类外的函数成员
    A(int x): a(x) { };
} a(5);
int f( ) { return a.a; }
int g( ) { return a.a; }
```
不正确，错误原因如上所述。

7. 完成下面堆栈类 STACK 和 REVERSE 类的函数成员定义。

```
class STACK {
    const int max;        //栈能存放的最大元素个数
    int   top;            //栈顶元素位置
    char *stk;
public:
    STACK(int max);
```

```
    ~STACK( );
    int push(char v);       //将v压栈，成功时返回1，否则返回0
    int pop(char &v);       //弹出栈顶元素，成功时返回1，否则返回0
};
class REVERSE: STACK {
public:
    REVERSE(char *str); //将字符串的每个字符压栈
    ~REVERSE( );          //按逆序打印字符串
};
void main(void) {
    REVERSE a("abcdefg");
}
```

函数成员定义如下：

```
STACK::STACK(int max) :max(max) {
    top = -1;
    if (max)
        stk = new char[max];
}

STACK::~STACK() {
    if (stk)
        delete[]stk;
    *(int*)&max = 0;
    top = -1;
}

int STACK::push(char v) {//将v压栈，成功时返回1，否则返回0
    if (top >= max - 1)
        return 0;
    stk[++top] = v;
    return 1;
}

int STACK::pop(char& v) {//弹出栈顶元素，成功时返回1，否则返回0
    if (top == -1)
        return 0;
    v = stk[top--];
    return 1;
}
```

```
REVERSE::REVERSE(char* str) :STACK(strlen(str)) {//将字符串的每个字符压栈
    for (int i = 0; i < strlen(str); i++)
        STACK::push(str[i]);
}

REVERSE::~REVERSE() {//按逆序打印字符串
    char temp;
    while (STACK::pop(temp)) {
        printf("%c", temp);
    }
    }
```

8. 找出下面的错误语句，说明错误原因。然后，删除错误的语句，指出类 A、B、C 可访问的成员及其访问权限。

```
class A {
    int a1;
protected:
    int a2;
public:
    int a3;
    ~A( ) { };
};

class B: protected A {
    int b1;
protected:
    int b2;
public:
    A::a1; 错误，A::a1 为 private，B 继承时无法访问，更无法改变其访问权限
    A::a2;
    int b3;
    ~B( ) { };
};

struct C: private B {
    int c1;
protected:
    int c2;
    B::A::a2;
    A::a3;
public:
    using B::b2;
    int c3;
    int a3;
    ~C( ) { };
};
```

```
int main() {
    C c;
    cout << c.b2;
    cout << c.B::b2; 错误，C 类对象 c 不能访问 B 的数据成员
}
```

错误语句与错误原因如上所述。删去后，类 A、B、C 可访问的成员及访问权限分别是：

类 A：

private：a1

protected：a2

public：a3，~A()

类 B：

private：b1

protected：A::a3，A::~A()，b2

public：A::a2，b3，~B()

类 C：

private：B::A::~A()，B::b3，B::~B()

protected：c2，B::A::a2，B::A::a3

public：B::b2，c3，a3，~C()，c1